

DCC Protocol and Troubleshooting

Contributed by [myndzi](#)

DCC Protocol and Troubleshooting (version 1.1!)

This tutorial will explain how DCC (Direct Client Connection) works, and explain how to troubleshoot DCC problems in mIRC. The information within is relevant to all IRC clients, however the methods and commands given are mIRC-specific; if you want to apply them to another client you will have to find the analog of these commands and settings on your own.

First off, the "inner workings" of DCC, such as they are.

Part 1 - DCC Protocol

A DCC connection begins with one client; we will call it Client A. Client A, which (in most cases) is connected to an IRC server, sets up a listening socket. It then sends a CTCP request to Client B, the recipient of the DCC connection. The CTCP request contains the type of connection, any relevant info (such as a filename and size), and Client A's IP address and the port it is listening on.

If Client B received the CTCP message, it then decides what to do about it; usually it can be configured to accept it, ignore it, or ask the user. If the user decides to accept the DCC request, Client B then attempts to establish a TCP connection to the IP and port given by Client A in the original CTCP message. If Client B establishes a connection, then the DCC transaction may continue.

Note that there is no message sent if Client B denies the request, therefore Client A must wait a certain amount of time and then give up. This is called a timeout. The amount of time varies, but need not be very long. Only in very extreme cases of lag can a short timeout cause a DCC to fail. (For example, if it takes longer than the timeout for the original CTCP to reach Client B, though this sometimes happens with XDCC servers that send data too fast.)

That is the basic concept. The actual CTCP messages follow these formats:

For DCC Chats: PRIVMSG ClientB : DCC CHAT Chat *longip port*

For DCC Sends: PRIVMSG ClientB : DCC SEND filename *longip port filesize*

There are some additional messages used when resuming a file, and also a couple alternative methods to establishing the DCC connection, but I won't go into those in this tutorial. See mIRC's excellent help file under the topics *DCC Resume Protocol*, *DCC Server Protocol*, and *DCC Socks5 Protocol* for information on these extensions of the DCC protocol.

The important thing to note here is that although Client A initiates the DCC connection, it is Client B that is actually doing the connecting. This means that to send a file or initiate a DCC Chat, Client A must be able to receive incoming TCP connections.

Part 2 - Troubleshooting DCC Problems in mIRC

Before we continue, note that commands are given in **bold**, parameters that you have to fill in are *italicized*, optional parameters are in **[brackets]** and choices are in **{braces | separated | by | pipes}** (use only one of the items listed).

First, connect to an IRC server.

Type **//dns \$me**

Then type **//echo -a \$ip**

If the ip you get from the //echo command matches **192.168.*.***, **10.*.*.***, or is between **172.16.0.0** and **172.31.255.255**, then skip the next two paragraphs.

If the two ip addresses are the same, then it is a firewall issue. If you are using Windows XP, try disabling the built-in firewall. If you are using Zone Alarm or Tiny/Kerio Personal Firewall, try checking that your mIRC

application is set to allow listening connections. If you are using another firewall, shame on you!

If the two addresses are not the same, type **/localinfo -u** (while connected!) and then type **//echo -a \$ip again**. If the ip you just echoed matches the one from the DNS, you may be good to go. Try to DCC Chat someone who is willing to help. Note that if you can establish a DCC Chat you can also establish a DCC Send; I want you to use DCC Chat for testing this issue because it helps us get to the bottom of the problem sooner.

If you *cannot* establish a DCC Chat connection, or your IP matched one of the previously listed ones, your issue is likely that you are behind a NAT (connection sharing) setup. Many DSL and Cable modems do NAT by default; you will have to read the manual to determine how to forward the ports. Once you know how, forward ports 113, 59, and 1500-1510 to your 'internal' IP address. Port 113 is for identd; some servers won't let you connect if you don't have it, and port 59 is the default DCC Server port. If a friend of yours isn't able to send files, you can type **/dccserver +s on 59** and he will be able to **/dcc send yourip** to send you files. You can get your 'internal' IP by running **winipcfg** from the start menu and looking at the IP address there, or if you run a server version of Windows (2000, NT, etc) you can drop to a command prompt and type **ipconfig**. If you don't like either of those methods, you can also type **/localinfo -h** followed by **//echo -a \$ip** in mIRC. Be sure to type **/localinfo -u** afterwards to set things back the way they should be.

A final note on NAT setups. If your computer is running Internet Connection Sharing for other computers in your house, it does *not* need to forward ports to itself. If you are in this situation and cannot send, investigate your firewall(s) more closely.

Once you have forwarded these ports, be sure to configure mIRC to use them. Go into mIRC options (alt+o), turn down the 'Connect' item, click 'Options', and click the 'Advanced...' button. and change the 'Port range for connections' section to read:

First: 1500

Last: 1510

Be sure that the 'DCC' checkbox is enabled, as well as the 'Use random ports' checkbox.

Note: Prior to mIRC 6.14, you would change the first and last ports by opening mIRC options, turning down 'DCC', and clicking 'Options'. The appropriate editboxes are labeled 'DCC Ports'.

After this is done, try testing a DCC Chat with a willing user again. If it succeeds, rejoice! If it doesn't, see the paragraph on firewalls, or double-check that your ports are forwarded properly (the correct ports, to the correct ip), and that you set the same port range in the DCC Options section that you forwarded in your router.

Note that because a DCC Chat succeeds, a DCC Send might still not. However, if DCC Chats work and DCC Sends don't, the problem is on the *recipient's* end; this means they need to either turn off DCC Ignore (in mIRC: **/dcc ignore off**), add you to their DCC Except list (**/dcc trust {nick/address}**), or on networks like Dalnet, perhaps type an additional command (such as **/dccallow +theirnick**).

You can, of course forward more ports than that if you wish, but you only need as many ports as you plan to have pending DCCs. This means that with 10 ports forwarded for DCCs, you can have 10 sends or chats waiting for a connection at once. Note that some scripts may listen on ports within the DCC range, and so long as they are listening on one of those ports it is unusable for DCCs.

Special case: If you are behind a NAT setup *and* you connect to IRC through a proxy (*bnc counts as a proxy), neither /localinfo method will do you any good. I thought there was an addon submitted to mirccscripts.org for this situation, but I can't seem to find it so I included some code at the end of this post. This code will not help you if you use a proxy to connect to websites as well.

If you are at work, school, a library, or otherwise unable to configure the NAT setup or the firewall that is blocking you from receiving incoming connections, there are two methods left to enable you to send files. However, both of them require that the person you are sending to can accept incoming connections. Method 1 is to use mIRC's DCC Server. The target recipient must type **/dccserver +s on port** where port is a port he can accept connections on, and defaults to 59. Then, you look up the person's IP address using **/dns nick** (or if the network you are on masks hosts, you need them to tell you their IP address, and hope they know it or how to find it) and finally type **/dcc send theirip [file]** where the [] around file mean it is optional; if you do not specify the filename mIRC will prompt you with a file select dialog box.

The second method has recently gotten a whole lot easier. As of mIRC version 6.17, Khaled has allowed explicit configuration of passive DCC sends. You can use **/dcc passive {on|off}** to enable and disable passive DCC connections. Using passive DCC requests essentially adds an extra parameter to a DCC message. If the

recipient supports passive DCC, it will then send you back a DCC message with its own IP address and a listening port, which you connect to. If you have no other options I highly recommend this method, but it should be noted that not all clients support it. Additionally, if the user you are sending to has misconfigured their own client, you will not be able to establish a connection. The best solution of course is a properly configured standard DCC send, but if all else fails, passive DCC makes a good second choice.

Part 3 - WhatIsMyIp.com lookup

Use this to correctly set your IP when you are behind a NAT network but using a bnc or other proxy.

```
; myndzi / v1.0
;
; whatismyip.com automatic local info configuration
; type /wimip to set up your local info; if your ip changes frequently
; (though i can't imagine why you'd need this if it did) you may want to
; put this in perform.

on *:start:.disable #wimhost

alias wimip sockclose wimip | sockopen wimip www.whatismyip.com 80

on *:sockopen:wimip:{
    if ($sockerr) return
    sockwrite -n $sockname GET / HTTP/1.1
    sockwrite -n $sockname Host: www.whatismyip.com
    sockwrite -n $sockname
}
on *:sockread:wimip:{
    if ($sockerr) return
    var %t | sockread %t
    while ($sockbr) {
        if (*<TITLE>*</TITLE>* iswm %t) {
            var %ip = $gettok(%t,4,32)
            localinfo $iif($host == $null,%ip,$host) %ip
            sockclose $sockname
            .enable #wimhost
            .timer 1 0 .dns $ip
            return
        }
        sockread %t
    }
}
on *:sockwrite:wimip:if ($sockerr) return
on *:sockclose:wimip:if ($sockerr) return

#wimhost off
on *:dns:{
    if ($dns(1).ip == $ip) .disable #wimhost
    if ($dns(1).addr != $null) localinfo $dns(%i).addr $ip
    echo -st Local host: $host ( $+ $ip $+ )
}
#wimhost end
```

All content is copyright by mirccscripts.org and cannot be used without permission. For more details, click [here](#).